ALGORITMOS Y ESTRUCTURAS DE DATOS III - Final

Fecha de examen: 03-MAR-2022

Apellido y nombre	L.U.	# hojas [1]
		4

Notas:

Ej1	Ej2	Ej3	Ej4	Final
B-	B	B-	B-	9
2,5	2,5	2	2	

1. (2.5 puntos) Dada una secuencia de n números naturales distintos, $S = \langle a_1, a_2, \dots, a_n \rangle$, se quiere determinar la longitud de la subsecuencia creciente de mayor longitud de S.

   Por ejemplo, si $S = \langle 9, 5, 2, 8, 7, 3, 1, 6, 4 \rangle$ las subsecuencias crecientes más largas son $\langle 2, 3, 4 \rangle$ y $\langle 2, 3, 6 \rangle$.

   a) Dar una fórmula recursiva que dada una secuencia S calcule la longitud de las subsecuencias crecientes más largas de S. Justificar su correctitud.

   b) Escribir un algoritmo de programación dinámica para resolver el problema.

   c) Calcular su complejidad.

2. (2.5 puntos) Decidir si las siguientes sentencias son Verdaderas o Falsas. Justificar:

   a) G conexo, tiene un único árbol generador sí y solo sí G es un árbol.

   b) Si la arista e es una arista puente de G, entonces e pertenece a todo árbol generador.

   c) Sea G un grafo con pesos en sus aristas. G tiene un único AGM sí y solo sí G es un árbol.

   F d) Si G tiene más de un AGM entonces toda arista que pertenece a algún AGM y no pertenece a algún otro AGM, está incluida en un circuito en el cual tiene peso mínimo. esa arista, no hay de mismo peso

3. (2.5 puntos) Un grafo k-partito completo es un grafo $G = (V, X)$ cuyo conjunto de vértices puede ser particionado en k subconjuntos $V_1, V_2, \dots, V_k$ ($\bigcup_{i=1}^{k} V_i = V$, $V_i \neq \emptyset\ \forall i$, $V_i \cap V_j = \emptyset$ si $i \neq j$), de modo que si $u \in V_i$ y $v \in V_j$, para $i \neq j$, entonces $(u, v) \in X$, y si $u, v \in V_i$ entonces $(u, v) \notin X$.

   B- a) ¿Cuál es el número cromático de un grafo k-partito completo?

   R b) Probar que el algoritmo goloso secuencial aplicado a un grafo k-partito completo produce un coloreo óptimo de G cualquiera sea el orden en que se tomen los vértices.

4. (2.5 puntos) Sean Π_1 y Π_2 dos problemas de decisión. Decir si las siguientes afirmaciones son verdaderas o falsas y justificar la respuesta:

   B a) ¿Qué se puede decir de Π_1 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_2 \in$ P?

   B b) ¿Qué se puede decir de Π_1 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_2 \in$ NP?

   B c) ¿Qué se puede decir de Π_1 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_2 \in$ NP-Completo?

   M d) ¿Qué se puede decir de Π_2 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_1 \in$ NP-Completo?

   B e) ¿Qué se puede decir de Π_2 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_1 \in$ NP-Completo y $\Pi_2 \in$ NP?

[1] Incluyendo esta hoja.

HOJA N° 01-04

FECHA 03/03/22

FINAL Algo. 3

(1) Dada una secuencia de n números naturales distintos, $S = \{a_1, \dots, a_n\}$, se quiere determinar la longitud de la subsecuencia creciente de mayor longitud de S.

Ej: $S = \{9, 5, 2, 8, 7, 3, 1, 6, 4\}$ Sols: $\{2, 3, 4\}$ y $\{2, 3, 6\}$.

a) Sea
$$f(i,j) = \begin{cases} S_i > S_j & \text{si } i == n-1 \\ f(i+1, j) & \text{si } i < n \wedge S_i < S_j \\ \max[f(i+1, i)+1, & \text{si } i < n \wedge S_i \geq S_j \\ \quad f(i+1, j), & \\ \quad f(i+1, j+1)] & \end{cases}$$
No es necesario

tomando la posición j como "el anterior" y ~~el~~ la posición i como el actual: El caso base ~~es~~ se da cuando i es igual a $len(S)^{-1}$, es decir, llego al final de la secuencia, ahí solo resta saber si para ese j dado, $S_i > S_j$, si lo es devuelve 1 ~~xx~~ (true), sino, 0. ✓

- Luego si ~~la pr posición~~ S_i es más chico que ~~la posición~~ S_j, continúo mirando la secuencia, ese valor no me aporta, por eso aumento i y dejo j como el anterior. ✓

~~Finalmente, si puedo sumar 1 a la cantid~~

- Finalmente si $S_i \geq S_j$, quiero elegir el máximo de las sigs. opciones:

.Utilizar S_i, por lo tanto ~~aumentar a~~ sumar +1 a f y seguir la recursión mirando lo que falta, con $j = i$, pues ahora S_i es el último anterior. ✓

NOTA

• No utilizar S_i (no aumentar f(i+1)), seguir mirando S con J el antecedente (o anterior) que ya estaba.

• No utilizar ~~S_i~~ y mover ambos índices, pero considerar una nueva subcadena sin los anteriores valores de S.

b. Guardo i y J en una matriz de longitud(s)-1 x longitud(s)-1.

M = [0 for i in range(len(s)-1)] for J in range(len(s)-1)]

↳ $M = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

(b)
```
def f(i, J):
    if (i == n-1):   / caso base.
        return S_i > S_j
    if (M[i][J] ≠ undefined):   // si ya lo calculé, lo devuelvo.
        return M[i][J]
    if (S_i < S_j):
        M[i][J] = f(i+1, J)
    else:
    M[i][J] = MAX( f(i+1, i)+1 , f(i+1, J) , f(i+1, J+1) )
    //            // uso S_i      // no uso S_i  // no uso S_i y avanzo J.
                                                    ?
    return M[i][J]
```

- print(f(0,0)).

(c) Dado que el algoritmo va guardando los valores ya calculados (y no recalcula), y recorre una matriz de n = longitud(s) - 1 por n y realiza comparaciones en O(1), El algoritmo tiene complejidad $O(n^2)$.

[termina pues siempre avanzo i]

HOJA Nº 02-04

FECHA

(2) a) G conexo, tiene un único AG $\Leftrightarrow$ G es árbol.

Verdadero

$\Rightarrow$) Supongo que no pasa que G es árbol, luego $\exists$ al menos un circuito simple. Entonces me construyo otro AG con alguna de las aristas de ese ciclo que no estaban en el "único AG" (HAY ALGUNA QUE NO ESTÁ EN EL AG, por ser árbol generador, Y SACANDO UNA DEL ÚNICO CICLO ~~existe~~). ABS que se formó pues G admite más de un AG y por HI tiene solo 1.

$\Leftarrow$) G es árbol y supongo que no tiene un único AG $\Rightarrow$ $\exists$ al menos una arista que está en T AG y no está en T' AG. Esta arista no es puente, por que de serlo estaría en todo AG. Entonces pertenece a un ciclo. ABS pues G es árbol y no tiene ciclos. ✓

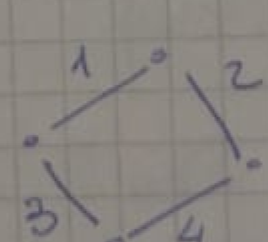
b) Verdadero: Supongo que e no pertenece a todo AG, pero entonces $\exists$ un T Árbol generador que no contiene a e, y por lo tanto: ~~T tiene más componentes conexas que G. ABS.~~

~~pues T es no conexo y~~ ... es puente que conecta al menos 2 cc.

T es no conexo porque no está e, y e es puente que conecta al menos 2 cc. ABS por T ser árbol conexo y generador.

c) F. G tiene único AGM $\Rightarrow$ G Árbol

Sea G



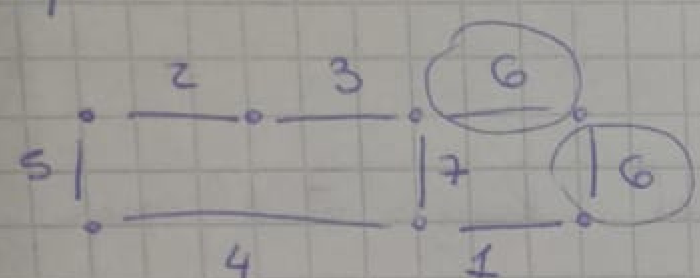
G tiene un único AGM por tener pesos distintos pero no es árbol.

✓

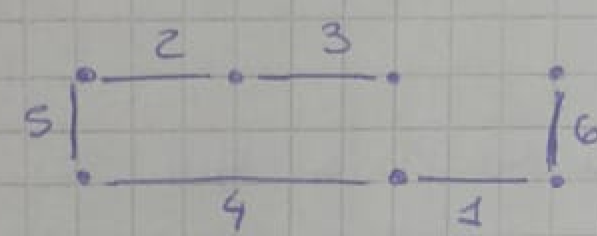
NOTA

(d). Falso. Contraejemplo:

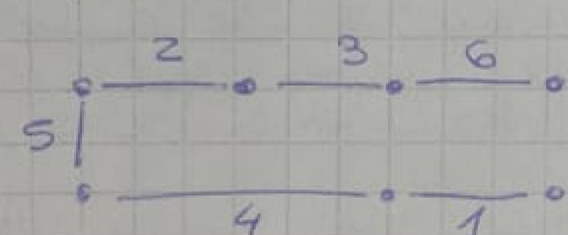
G :



1. AGM :



2. AGM :



Las aristas de valor 6, pertenecen: 1 al AGM 1 y la otra al AGM 2, sin embargo no son de peso mínimo dentro de su ciclo, pues existe la arista con peso/valor 7.

HOJA Nº 03-04

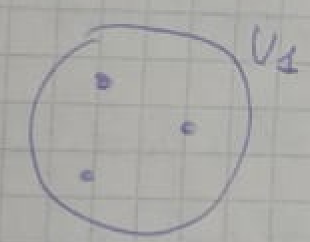
FECHA

(3) Sea G un grafo k-partito completo:

a) ¿Cuál es el número cromático de G? ~~ese grafo~~

Veamos: • 1-k-partito completo

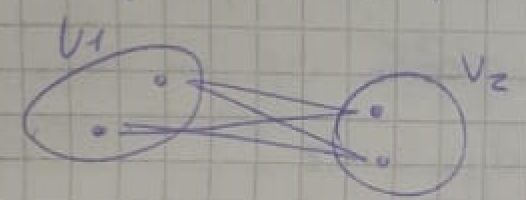
es el grafo de la pinta:



G = nodos aislados.

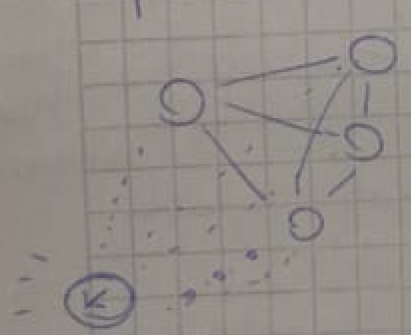
Solo necesito 1 color.

• 2-k partito completo:



Necesito al menos 2 colores, pues cada vértice se conecta con todos los otros de las otras particiones menos la suya.

Entonces necesitaría k colores, mínimo, pues puedo colorear k-1 particiones con k-1 colores.



pero si tuviera una partición más, y estuviera conectada a las k-1 otras particiones, para no repetir colores adyacentes necesitaría 1 color más.

ESTÁS ASUMIENDO QUE EN UN COLOREO ÓPTIMO DE G, TODO SUBGRAFO ESTÁ COLOREADO DE FORMA ÓPTIMA Y ESO NO ES CIERTO. EN ESTE CASO VALE, PERO HAY QUE JUSTIFICARLO

NOTA

b) Probar que el algoritmo goloso secuencial aplicado a un grafo k-partito completo produce un coloreo óptimo de G cualquiera sea el orden en que se tomen los vértices.

Supongamos que partimos del vértice v_1. V_1 es adyacente a todo nodo de otra partición menos la suya luego utiliza el menor color posible "1" y se mueve a alguno de sus adyacentes. ¿Por qué? Nuevamente, este nodo V_2 es adyacente al v_1 de color 1 y a todos los otros de las otras particiones, menos la suya, luego no le queda otra ~~opció~~ opción que tomar el color "2" y moverse a un adyacente:

Si este adyacente, no es vecino de alguno que ya tiene color, puede reutilizar algún color; sino toma uno nuevo.

pues nunca va a compartir color por ser k-partito completo.

Luego el algoritmo termina reutilizando colores de forma válida o tomando 1 nuevo (por partición) si no le quedaba otra opción, usando k colores, por k particiones. } Esto es lo que hay que demostrar

Entonces produce un coloreo óptimo y como no supusimos nada de v_1, lo hace desde cualquier vértice.

Si supiéramos que no da el óptimo, ∃ algoritmo que colorea a G con k-1 colores o menos. y G lo hace en k, pero por a) el núm. crom. de G es k, luego el algoritmo goloso da el óptimo (pues no existe coloreo válido con k-1 o menos colores).

¿Por qué no con más?

Reg

HOJA Nº 04-04

FECHA

4 a) ¿Qué se puede decir de Π_1 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_2 \in P$?

Sabemos que existe f función polinomial que transforma instancias de Π_1 en instancias de Π_2 y que $\Pi_2 \in P$, es decir, existe algoritmo polinomial que lo resuelve. Luego ~~Π_1~~ puedo transformar una instancia de Π_1 a una instancia de Π_2 y resolverla en tiempo polinomial.

$\Rightarrow \Pi_1 \in P$.

b) ¿Qué se puede decir de Π_1 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_2 \in NP$?

~~Por la reducción p~~

Sabiendo que existe f polinomial que transforma instancias de Π_1 en instancias de Π_2, ~~es decir~~ y que $\Pi_2 \in NP$, "Π_1 no puede ser más difícil que Π_2", ~~pues podemos transformar la instancia y resolverla~~ ya que de existir algoritmo que resuelve Π_2 podría utilizarlo para resolver $f(I_{\Pi_1})$, luego, $\Pi_1 \in NP$.

NOTA

c) ¿Qué se puede decir de Π_1 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_2 \in$ NP-Completo?

Sabemos que $\Pi_2 \in NP$ y $\Pi_2 \in$ NP-Hard y sabemos que existe una ~~reducción~~ función polinomial que transforma instancias de Π_1 en instancias de Π_2, es decir, sabemos que $\Pi_1 \in NP$, pero no podemos asegurar que pertenezca a NP-Hard o no. Por lo tanto no podemos asegurar que $\Pi_1 \in NPC$ o no. (por mismo argumento que en (b))

d) ¿Qué se puede decir de Π_2 sabiendo que existe una reducción polinomial de Π_1 a Π_2 y que $\Pi_1 \in$ NP-Completo?

Si suponemos $P \neq NP$, tenemos el sig. gráfico:

. Al saber que existe reduc. pol. de Π_1 a Π_2 y que $\Pi_1 \in NPC$, sabemos que Π_2 es "al menos tan difícil que Π_1" pero no podemos asegurar que exista un certificado que pueda comprobar Π_1 en tiempo polinómico, es decir, no podemos asegurar que $\Pi_1 \in NP$. Por lo tanto, tampoco podemos asegurar que $\Pi_1 \in NPC$.

$\Pi_1 \in NP\text{-}C$, lo que implica que $\Pi_1 \in NP$.

Se puede asegurar que $\Pi_2 \in$ NP-Hard.

e) ¿Qué ~~puede~~ se puede decir de Π_2 sabiendo que existe una reduc. pol. de Π_1 a Π_2 y que $\Pi_1 \in$ NP-Completo, y $\Pi_2 \in NP$?

Sabiendo que se cumplen las condiciones del teorema de Cook-Levin, es decir $\exists\, \Pi_1 \in NPC$ que se puede reducir a Π_2 y $\Pi_2 \in NP$, podemos asegurar que $\Pi_2 \in NPC$.